4.5. All to all communication on a balanced binary tree with $p$ processors
processors at leaf nodes
Links are bidirectional
An exchange of two $m$-word messages between any two nodes takes
$t_1 + t_2 \cdot m \cdot k$
if the link is carrying $k$ simultaneous messages
Remember that as we go up the tree, the links carry more messages
you can view this also as the size of the message increasing

Homework #3 Due Feb. 13, 2019: 4.5, 4.7, 4.12, 4.14, 4.19
4.7. One to all personalized communication \( \rightarrow \) scatter
one node sends different messages to different nodes
So each intermediate node keeps its message and passes rest of the messages

4.12. Very minor change. There are no wrap-around connections in a linear array (becomes a ring with wrap around) and 2-D messages. You may need to travel farther

4.19 The problems includes a hint

<table>
<thead>
<tr>
<th>f</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.10</td>
<td>0.77</td>
<td>0.59</td>
<td>0.40</td>
<td>0.24</td>
<td>0.14</td>
</tr>
<tr>
<td>0.15</td>
<td>0.69</td>
<td>0.49</td>
<td>0.31</td>
<td>0.18</td>
<td>0.10</td>
</tr>
</tbody>
</table>

**Gustavson’s Law.** Here instead of comparing the execution times we will try to see how much *more work* a multiprocessing system can perform compared a single processor in the same amount of time
CSCE 5160 Parallel Processing

Work done by single processor in \( T_1 = W = W^*f + W^*(1-f) \)
Work done by \( p \) processors = \( Wp = W^*f + p^*[W^*(1-f)] \)
More work done as we increase \( p \)

<table>
<thead>
<tr>
<th>( f )</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
<th>1000000</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.10</td>
<td>3.70</td>
<td>7.30</td>
<td>14.50</td>
<td>28.90</td>
<td>57.70</td>
<td>900000.10</td>
</tr>
<tr>
<td>0.15</td>
<td>3.55</td>
<td>6.95</td>
<td>13.75</td>
<td>27.35</td>
<td>54.55</td>
<td>850000.15</td>
</tr>
<tr>
<td>0.5</td>
<td>2.50</td>
<td>4.50</td>
<td>8.50</td>
<td>16.50</td>
<td>32.50</td>
<td>500000.50</td>
</tr>
</tbody>
</table>

Iso-Efficiency
How much more work should \( p \) processor do to keep the same efficiency

Overhead in executing parallel algorithms \( T_o(W,p) \) \( \rightarrow \) depends on \( p \) and \( W \)

Execution time with \( p \) processors \( T_p = [W + T_o(W,p)] / p \)

Speed up \( S = \frac{W}{T_p} = \frac{(W^*p)}{[W + T_o(W,p)]} \)

Efficiency \( E = \frac{S}{p} = \frac{1}{[1 + T_o(W,p)/W]} = W/[W+T_o(W,p)] \)

Let us examine our previous example of adding \( n \) numbers

serial complexity = \( n = W \)

Execution time with \( p \) processors = \( \frac{(n/p)}{+2*\log(p)} = \frac{W/p +2*\log(p)}{p} \)

What is the overhead part?

\( T_o(W,p) = 2* p^*\log (p) \)

Our iso-efficiency function will be \( K*2^*p^*\log(p) \)
Let us plot this function for different efficiency levels
These numbers are from the Iso-efficiency equation. Let us compare them from the data we have seen before by using $T_p$ and efficiency equations

<table>
<thead>
<tr>
<th>$n$</th>
<th>$p=1$</th>
<th>$p=4$</th>
<th>$p=8$</th>
<th>$p=16$</th>
<th>$p=32$</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>1</td>
<td>0.8</td>
<td>0.57</td>
<td>0.33</td>
<td>0.17</td>
</tr>
<tr>
<td>192</td>
<td>1</td>
<td>0.92</td>
<td>0.8</td>
<td>0.6</td>
<td>0.38</td>
</tr>
<tr>
<td>320</td>
<td>1</td>
<td>0.95</td>
<td>0.87</td>
<td>0.71</td>
<td>0.5</td>
</tr>
<tr>
<td>512</td>
<td>1</td>
<td>0.97</td>
<td>0.91</td>
<td>0.8</td>
<td>0.62</td>
</tr>
</tbody>
</table>

We also analyzed Matrix multiplication with $p$ processors connected as a 2-D mesh

Each processor needs the columns assigned to the all other processors.

We can use *all to all* broadcast on 2-D mesh of $p^{1/2} \times p^{1/2}$

Communication cost is $= (p-1)n^2/p = O(n^2)$

Computation cost $= 2(n) * (n^2/p) = 2 * n^3/p$

Total Execution time $= T_p = (n^3) + 2 * n^3/p = [n^3 + (n^2 * p)/2] / p = [W + T_o(W,p)] / p$

Note for matrix multiplication, $W = n^3$

$T_o(W,p) = (n^2 * p)/2 = (W^{2/3} * p)/2$

The isoefficiency function is $K * T_o(W,p) = K * (W^{2/3} * p)/2$

remember $K = (E/1-E)$
CSCE 5160 Parallel Processing

The isoefficiency function is 
\[ K^* T_o(W,p) = K^* \left( \frac{W^{2/3} + p}{2} \right) \]
\( \text{remember } K = \left( \frac{E}{1-E} \right) \)

\[ \text{Speedup } = \frac{W}{T_p} = p \left[ W + \left( \frac{W^{2/3} + p}{2} \right) \right] \]
\[ E = \frac{\text{Speedup}}{p} = 2 \left[ \frac{2 + pW^{1/3}}{2} \right] \]
\[ \text{Or } W = \left( \frac{pE}{2 - E} \right)^3 \]

<table>
<thead>
<tr>
<th>E</th>
<th>Work (Processors)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4</td>
</tr>
<tr>
<td>0.6</td>
<td>5.04</td>
</tr>
<tr>
<td>0.75</td>
<td>13.82</td>
</tr>
<tr>
<td>0.8</td>
<td>18.96</td>
</tr>
<tr>
<td>0.9</td>
<td>35.05</td>
</tr>
</tbody>
</table>

CSCE 5160 Parallel Processing

Other Scalability Measures

Scaled Speedup – Speedup calculation based on increased problem size
In other words, we will \textit{linearly} increase problem size with \# processors and compute speedup

\[ \text{Scaled Speedup} = \frac{\text{Speedup}}{f(W,p)} = \frac{T_1}{T_p(p,W)} \]

If we have \( p \) processors, we will estimate the time to complete a larger problem proportional to \( p \) when determining \( T_p \)

Textbook uses \( k*p*W \) for increasing work proportional to \# processors

Consider applying this with Amdahl’s law
I am going to use \( p*W \) for work for \( p \) processors.

\[ T_{\text{scaled}} = (W^p*p)^f + (W^p*p)^{(1-f)} = W^p + (1-f)W \]

Scaled-Speedup = \( \frac{W}{T_{\text{scaled}}} = \frac{1}{[pf + (1-f)] = 1/[1+(p-1)f]} \)

Original = \( p/[1+(p-1)*f] \)
CSCE 5160 Parallel Processing

Memory Scaled: When we talked about scaling work proportional to p (processors), how do we come up with work.

Consider the following. We will assume that with p processors, the available memory will be p times as large as one processor. Can we scale the work proportional to available memory?

Example: 5.21 on page 223 of Textbook

Consider matrix * vector product \( A \times x = b \)

On a single processor: \( T_1 = t_c \times n^2 \)

where \( t_c \) is the computational time for a single Multiply and Accumulate computation.

Note: Scaled speedup <1 since we increased work with processors
CSCE 5160 Parallel Processing

\[ T_p = n^2/p + 2t(p^{1/2} - 1) + t_o n \]

To simplify, let us set all times to be the same and set them = 1
\[ T_p = n^2/p + 2(p^{1/2} - 1) + n \]

Note \( W = n^2 \) or \( n = W^{1/2} \)
\[ T_p = W/p + [2(p^{1/2} - 1) + W^{1/2}] = (W + p^n [2(p^{1/2} - 1) + W^{1/2}])/p \]

So, \( T_0 = p^n [2(p^{1/2} - 1) + W^{1/2}] \rightarrow W^{1/2} = n \)

Speedup \( S_{\text{original}} = W/T_p = W/[W/p+2(p^{1/2} - 1)+W^{1/2}] = W^p/[W+W^{1/2}p+2^p p^{1/2} - 2] \)

Now we can do Iso-efficiency. But let us return to scaled speedup with memory

What does that mean for this problem?

if we double work to 2 \( W \), what is the value of new \( n \)?
\[ 2^{1/2} n. \text{ If we set work to } pW = p^n n^2 \]
\[ \text{new } n = p^{1/2} n \]

CSCE 5160 Parallel Processing

\[ T_{p,\text{scaled}} = (\text{new } n^2)/p + 2(p^{1/2} - 1) + (\text{new } n) \]
\[ = (p^{1/2}n^2)/p + 2(p^{1/2} - 1) + p^{1/2}n = (p^n n^2)/p + 2(p^{1/2} - 1) + p^{1/2}n \]
\[ n^2 + (n+2)p^{1/2} - 2 = W + (W^{1/2}+2)p^{1/2} - 2 \]
\[ = [(Wp + (W^{1/2}+2)p^{1/2})]/p \]
\[ T_{\text{new-overhead}} = (W^{1/2}+2)p^{1/2} \]

Ignore some terms \( T_p = W + W^{1/2} p^{1/2} \)

Speedup \( S_{\text{scaled}} = W/[W + (W^{1/2}+2)p^{1/2} - 2] = 1/[1+p^{1/2}(W^{1/2}+2)] \)

Compare this with Speedup \( S_{\text{original}} = W^p/[W+W^{1/2}p+2^p p^{1/2} - 2] \)

No \( p \) in the numerator in scaled version

Solve similar problem for Matrix multiplication

Remember if we increase work, the amount of memory increase as \( n^2 \)
There is no change since Matrix \( \times \) vector and Matrix \( \times \) Matrix
memory scales the same way

(the communication costs may be different)
One more example of scaled speed up. Problem 5.5 on page 230.

Here we are looking at adding n numbers using p processors
It takes 10 time units to communicate data between two processors and 1 time unit to add numbers.

Consider adding n numbers \[ \text{for } (i=0; i<n; i++) \text{ sum } = \text{sum } + a[i]; \]

Sequential time = \( t_a \cdot n \)

In the optimal algorithm, we view the computation as a binary tree of adders
the total execution time using \( p \) processors = \( (t_a \cdot n/p) + (t_c + t_{\text{comm}}) \text{log } (p) \)

Let us assume \( t_a = 1 \) and \( t_{\text{comm}} = 10 \)

\[ T_p = (n/p) + 11* \text{log}(p) \]

Scaling here will be \( n = k \cdot p \)

\[ \text{Scaled speed } = k/[11* \log p] \]

CSCE 5160 Parallel Processing

CSCE5160 February 6, 2019
CSCE 5160 Parallel Processing

Consider problem 5.9  Here we are looking at time constrained scaling (Gustavson’s)

If it takes $T_1$ for one processor to complete $W$ work, how much work can we do in $T_1$ using $p$ processors

Note we are to use the expression from problem 5.5: $T_p = \frac{n}{(p-1)} + 11 \log p$

We need to limit the time $T_p$ to 512

We now solve for $n$ (which is the work on single processor) for different $p$

<table>
<thead>
<tr>
<th>$p$</th>
<th>1</th>
<th>4</th>
<th>16</th>
<th>64</th>
<th>256</th>
<th>1024</th>
<th>4096</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$</td>
<td>512</td>
<td>1964</td>
<td>7504</td>
<td>28608</td>
<td>412672</td>
<td>1560576</td>
<td>40961560576</td>
</tr>
</tbody>
</table>

One final example. Consider Quicksort(list)

Split_list(list, pivot, list1, list2);
Quicksort(list1);
Quicksort(list2);

Recursive partitioning: Initially one task, then 2, then 4, …. Total number of tasks = $1+2+…+\log(n) = 2^{\log(n)}-1$

Communication: first step: list1 and list2 (2 messages of size $n/2$) then 4 messages (but can be in parallel) of size $n/4$

Think about how you would express the execution times

Computational: $(n+n/2+…+2)t_c$

If no communication conflicts and messages travel just one node communication cost: $(n/2 +…+2)t_c$

We will see a different version of parallel quicksort later

Read Chapter 5. The next homework will be from this material